ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

BRUNNA FERNANDES PAVIN

RAFAEL BENTO MARIANO DA SILVA

SOLUÇÃO DE GERENCIAMENTO DE ESTACIONAMENTO EM VIAS URBANAS EM CENTROS COMERCIAIS BASEADAS EM MACHINE LEARNING

Catanduva-SP

2021

Elaborada pela biblioteca do IFSP Câmpus Catanduva
com dados fornecidos pelo autor.

Pavin, Brunna Fernandes

P338s

Solução de gerenciamento de estacionamento em vias urbanas em centros comerciais baseadas em machine learning / Brunna Fernandes Pavin, Rafael Bento Mariano da Silva. – Catanduva, SP: IFSP, 2021.
33 f.

Orientador: Márcio Andrey Teixeira

Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) -- Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Catanduva, 2021.

1. Estacionamento. 2. Mobilidade urbana. 3. Machine learning. 4.Carros. I. Silva, Rafael Bento Mariano da. II. Teixeira, Márcio Andrey. (Orient.). II. Título.

CDD (23. ed.): 006.31

BRUNNA FERNANDES PAVIN
RAFAEL BENTO MARIANO DA SILVA

SOLUÇÃO DE GERENCIAMENTO DE ESTACIONAMENTO EM VIAS URBANAS EM CENTROS COMERCIAIS BASEADAS EM MACHINE LEARNING

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, campus Catanduva como requisito parcial para conclusão do curso de Análise e Desenvolvimento de Sistemas.
Orientador: Prof. Dr. Márcio Andrey Teixeira

Catanduva-SP
2021

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
São Paulo

Ministério da Educação
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo
Câmpus Catanduva
COORD CURSO TEC INTEGRADO DE INFORMATICA

ATA N.º 15/2021 - TIF-CTD/DAE-CTD/DRG/CTD/IFSP

Ata de Defesa de Trabalho de Conclusão de Curso - Graduação

Na presente data realizou-se a sessão pública de defesa do Trabalho de Conclusão de Curso intituladoSOLUÇÃO DE GERENCIAMENTO DE ESTACIONAMENTO EM VIAS URBANAS EM CENTROS COMERCIAIS BASEADAS EM MACHINE LEARNING apresentado(a) pelos(a) alunos(a) Brunna Fernandes Pavin (CT3001288) e Rafael Bento Mariano da Silva (CT1710907) do Curso SUPERIOR EM TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS, (Câmpus Câmpus Catanduva). Os trabalhos foram iniciados às 21:00 pelo(a) Professor(a) presidente da banca examinadora, constituída pelos seguintes membros:

Membros	IES	Presença (Sim/Não)	Aprovação/Conceito (Quando Exigido)
Márcio Andrey Teixeira (Presidente/Orientador)	IFSP	Sim	9.0
Luis Hideo Vasconcelos Nakamura (Examinador 1)	IFSP	Sim	9.0
Murilo Secchieri de Carvalho (Examinador 2)	IFSP	Sim	9.0

Observações:

A banca examinadora, tendo terminado a apresentação do conteúdo da monografia, passou à arguição dos(a) candidatos(a). Em seguida, os examinadores reuniram-se para avaliação e deram o parecer final sobre o trabalho apresentado pelos(a) alunos(a), tendo sido atribuído o seguinte resultado:

[X] Aprovado(a) [ ]Reprovado(a) Nota (quando exigido): 9.0

Proclamados os resultados pelo presidente da banca examinadora, foram encerrados os trabalhos e, para constar, eu lavrei a presente ata que assino juntamente com os demais membros da banca examinadora.

Câmpus Catanduva, 29 de novembro de 2021

Avaliador externo: [ ]Sim [X]Não

Assinatura:
ATA N.º 15/2021 - TIF-CTD/DAE-CTD/DRG/CTD/IFSP

Documento assinado eletronicamente por:

- Marcio Andrey Teixeira, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 29/11/2021 22:45:45.
- Rafael Bento Mariano da Silva , CT1710907 - Discente, em 29/11/2021 22:49:22.
- Luis Hideo Vasconcelos Nakamura, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 29/11/2021 22:57:35.
- Murilo Secchieri de Carvalho, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 29/11/2021 23:20:55.
- Brunna Fernandes Pavin, CT3001288 - Discente, em 30/11/2021 20:45:18.

Este documento foi emitido pelo SUAP em 29/11/2021. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 264088
Código de Autenticação: 0706823262

RESUMO

Este projeto aborda um dos principais problemas da sociedade moderna, a mobilidade urbana, tendo como foco a mobilidade em centros comerciais que outrora consistiam-se em pequenos grupos de mercadores em busca de trocas de mercadorias, tornou-se o que conhecemos hoje e denominamos “centros urbanos “. Devido a expansão muitas vezes desenfreada dos centros urbanos, nos deparamos com questões problemáticas acerca de encontrar um local para estacionar o veículo de forma rápida e eficiente. A solução proposta nesse trabalho utiliza soluções baseadas em *machine learning* para melhorar o problema existente em encontrar uma vaga de estacionamento de forma mais rápida e eficiente, sendo este, um problema acerca do tema abordado em mobilidade urbana. Através da captação de imagens por meio de câmeras de segurança, ccri-se um algoritmo de aprendizagem de máquina que seja capaz de reconhecer características especificas em objetos que compõem um estacionamento. Isso será possível graças a um treinamento prévio supervisionado no algoritmo. Desta forma, pretendemos com a solução proposta facilitar a procura de uma vaga de estacionamento disponível.

Palavras-chave: Estacionamento, mobilidade urbana, *machine learning*, *deep learning,* python, *OpenCV*, carros.

ABSTRACT

This project addresses one of the main problems of modern society, urban mobility, focusing on mobility in shopping centers that once consisted of small groups of merchants seeking commodity exchanges, has become what we know today and call “urban centers ". A major contributing factor to this was the industrial revolution that took place around the eighteenth century, the machines then assumed the essential role in favor of human evolution, since then the popularity of these machines has only grown to extraordinary levels, and the machine that we will approach the The bottom line in this article will be the motor vehicle, the main mode of transportation today. Due to the often-unbridled expansion of urban centers, we are faced with problematic questions about space and how to manage it efficiently. The solution described below tries to use machine learning technology and solutions, to solve a problem about the theme, by capturing images through security cameras. We intend to create a machine learning algorithm that can recognize Specific characteristics in objects that make up a parking lot, this will be possible thanks to a previous supervised training in the algorithm, we intend with the solution to facilitate the task of finding a parking space available.

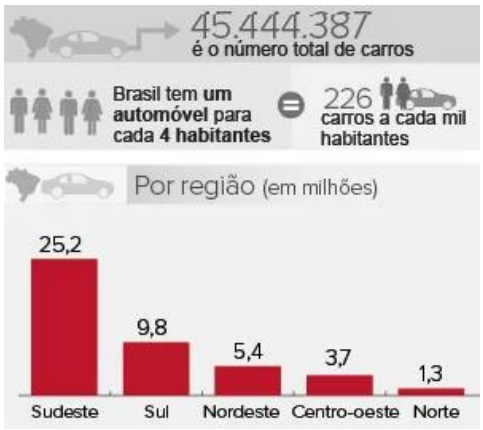
Keywords: Parking, urban mobility, *machine learning, deep learning, python, OpenCV*, cars.

LISTA DE FIGURAS

1.INTRODUÇÃO

Atualmente os grandes centros urbanos possuem um grande fluxo de veículos, muito dos quais poderiam ser evitados com atitudes relacionadas ao uso da informação. Um problema corriqueiro é a procura por vagas de estacionamento disponíveis nas vias urbanas, que indiretamente afeta o fluxo dos veículos que transitam nas vias, consequentemente atrapalhando ainda mais um problema cabal enfrentado pela gestão pública. Os grandes centros urbanos já não comportam o crescimento populacional diretamente atrelado a crescente do número de veículos. Segundo reportagem do Portal de Notícias da Globo [G1], com o aumento da frota, o Brasil já tem um automóvel para cada 4,4 habitantes. São 45,4 milhões de veículos do tipo. Há dez anos, a proporção era de 7,4 habitantes por carro. Com base nos números de registros do Departamento Nacional de Trânsito (Denatran) e nas estimativas populacionais do Instituto Brasileiro de Geografia e Estatística (IBGE) de 2013 revela que, das dez cidades com mais carro por habitante, nove estão na região Sudeste.

Figura 1 - Frota de veículos.

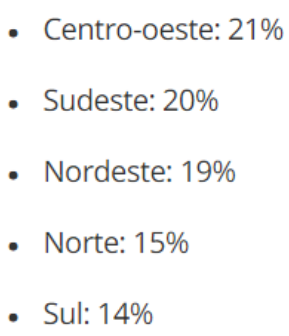


Fonte: Thiago Reis. Com aumento da frota, país tem 1 automóvel para cada 4 habitantes, 2014,**G1** Disponível em: http://g1.globo.com/brasil/noticia/2014/03/com-aumento-da-frota-pais-tem-1-automovel-para-cada-4-habitantes.html. Acesso em 26 de março de 2021.

Soluções integradas com a utilização da informação como ferramenta essencial têm como papel amenizar problemas enfrentados atualmente nos centros urbanos, entretanto é possível observar uma enorme defasagem no território nacional. Dados alarmantes coletados pelo Centro Regional de Estudos para o Desenvolvimento da Sociedade da Informação [Cetic]

apontam que atualmente apenas 18 por cento das prefeituras brasileiras possuem planos de "cidades inteligentes".

Figura 2 - Percentual de municípios por região com plano de cidade inteligente.



Fonte: Helton Simões Gomes, Só 18% das prefeituras têm plano de cidade inteligente no Brasil, diz pesquisa, **G1**, 2018 Disponível em: https://g1.globo.com/economia/tecnologia/noticia/so-18-das-prefeituras-tem-plano-de-cidade-inteligente-no-brasil-diz-pesquisa.ghtml Acesso em 20 de março de 2021.

Como visto na imagem acima, os municípios têm em mente tornar as cidades inteligentes fazendo com que a utilização de um mapeamento do território urbano possível, através de processamento de imagem, desenvolver uma aplicação que possa informar um usuário que estaria à procura de uma vaga, direcionando-o a um local já vago eliminando assim a necessidade do mesmo transitar a via a procura de um local. Utilizando esta aplicação é possível diminuir o tráfego de veículos nas vias, consequentemente, diminuindo a eliminação das emissões de gases atrelados a queima do combustível e consequentemente seus problemas e agilizando a locomoção do usuário.

Contudo o objetivo deste trabalho tem como objetivo auxiliar os condutores de um centro urbano que estão à procura de uma vaga de estacionamento. Para isso, utilizaremos conceitos de *Machine Learning* (ML) como também conceitos de Computação Visual. Através da captação de imagens por um dispositivo de captura, serão aplicadas operações morfológicas nestes dados coletados com o propósito de auxiliar no posterior processamento das mesmas. Neste caso, utilizaremos uma biblioteca chamada *OpenCV* [OCV] que será descrita em seção posterior deste documento. Após aplicar filtros nas imagens geradas pelo dispositivo de

captura, essas imagens serão submetidas ao processamento de um algoritmo de *machine Learning*, que tem como principal função verificar se a imagem gerada pela biblioteca *OpenCV* é de um automóvel, como também verificar a localização deste automóvel no espaço de estacionamento previamente definido. Isso será feito graças a inúmeros testes e treinamentos que serão realizados para criar um modelo de *machine Learning* que irá verificar o "status" final sobre o estado da vaga a ele submetido, sendo este "status" ocupado ou livre. Tal informação será atualizada em uma base de dados que irá disponibilizar esta informação para uma aplicação móvel, que por sua vez irá mostrar para o usuário final se as vagas de estacionamento coberta pela aplicação estão livres ou ocupadas.

A ideia central desse trabalho é facilitar na parte de conseguir obter uma vaga por meio de uma inteligência artificial. E que seja capaz de mostrar ao seu usuário onde essas vagas desocupadas estão, facilitando assim a vida de motoristas por meio da tecnologia. Assim sendo, o objetivo principal é desenvolver uma aplicação capaz de identificar vagas de estacionamento ociosas, sem a necessidade de interferência humana, visando à otimização do tempo do usuário que a utilizar. Com isso, destaca-se como objetivos específicos:

- Auxiliar condutores de veículos a encontrarem vagas de estacionamento, sem a necessidade de uma varredura total ou parcial de destino.
- Evitar trânsito desnecessário proveniente de uma busca desenfreada conduzida por motoristas a procura de vagas disponíveis.
- Abstrair a necessidade de um indivíduo checar o estado de uma vaga de estacionamento, está função será realizada por um algoritmo de ML.
- Obter através de um dispositivo de captura de imagens, dados que serão submetidos a operações em uma biblioteca chamada *OpenCv* de modo que auxilie no posterior processamento.
- Utilizar um algoritmo com propriedades de aprendizado de máquina para torná-lo capaz de detectar o estado de uma vaga de estacionamento, dada uma imagem como parâmetro inicial.
- Utilizar bibliotecas como a *Opencv* para, a partir da detecção, delimitar se a vaga de estacionamento está ou não ocupada.

O restante dos capítulos da presente monografia está organizado da seguinte forma: No Capítulo 2 será apresentada as tecnologias que foram utilizadas para dar procedimento ao projeto apresentado e suas métricas de desempenho.

No Capítulo 3 serão mostrados os trabalhos relacionados que foram úteis para toda a ideia do projeto em si.

O Capítulo 4 apresenta todo o desenvolvimento, desde a aplicação, até a parte do reconhecimento, entre outros.

No Capítulo 5 contém as conclusões e considerações finais sobre todo o projeto apresentado.

O Capítulo 6 destina-se referências bibliográficas utilizadas para a pesquisa.

2. TECNOLOGIAS ENVOLVIDAS

Neste capítulo descreveremos as tecnologias utilizadas na elaboração do projeto.

2.1. OpenCV

Neste projeto foi utilizado a biblioteca *open source* denominada *OpenCV* [OCV]. Esta biblioteca foi originalmente apresentada no ano de 1999, como sendo uma proposta de avanço na área de estudos no campo de visão computacional. O projeto foi inicialmente desenvolvido pela Intel e consiste em códigos criados na linguagem de programação C/C++, auxiliando desenvolvedores que pretendiam utilizar em suas aplicações recursos de processamento de imagens. Apesar do desenvolvimento do *OpenCV* ser feito nas linguagens de programação C/C++ ele também oferece suporte a diversas outras linguagens, tais como: Java e Python. Neste projeto será utilizada a linguagem Python em conjunto do *OpenCV.*

2.2. Linguagem Python

A linguagem de programação escolhida para o desenvolvimento do projeto proposto é o Python [PY]. Projetada no ano de 1991 por Guido van Rossum, o Python possui atualizações constantes até os dias de hoje, essa linguagem tem como filosofia enfatizar a importância do esforço do programador sobre o esforço computacional, priorizando a legibilidade do código combinando-a com uma sintaxe concisa e clara com os recursos poderosos de sua biblioteca padrão, módulos e *frameworks* que consistem de softwares que tem como finalidade ajudar o desenvolvimento com ferramentas prontas (Noleto, 2020), quanto aos requisitos técnicos Python é uma linguagem de alto nível, multiplataforma que suporta vários paradigmas tais como, orientado a objetos, imperativo, funcional e procedural, outra vantagem é a baixa extensão dos códigos necessários para realização de uma tarefa se comparado a outras linguagens, ela é principalmente utilizada para processamento de textos e dados científicos.

2.3. Machine Learning

Consiste na realização de uma tarefa por um sistema computacional sem a utilização de instruções explicitas de programação. Para que isso ocorra, o algoritmo é submetido ao que denominamos de “treinamento”, também conhecido como “*training data* “.

Um algoritmo de *machine learning* pode ser treinado das seguintes maneiras: Aprendizagem supervisionada, aprendizagem não supervisionada. No treinamento supervisionado utilizamos um conjunto de dados composto de imagens previamente rotuladas a qual classe determinados objetos pertencem, ou seja, mostramos de maneira explicita que se o algoritmo futuramente detectar algo parecido com as características do que rotulamos ele as classifique como foi proposto.

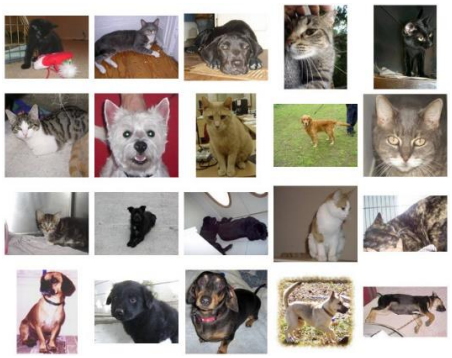
Figura 3 – Exemplo de conjunto de imagens de Motos.

Fonte: Illumination Estimation and Cast Shadow Detection through a Higher-order Graphical Model. **Researchgate**, 2011 Disponível em: < https://www.researchgate.net/figure/Results-for-the-Motorbikes-class-of-the-Caltech101-dataset-A-synthetic-sun-dial-orange_fig4_224254887 >Acessado em 27 de março de 2021.

Como visto na figura 3 é possível ensinar ao algoritmo a distinguir como é um determinado objeto, assim facilitando na hora de detectá-lo.

A diferença da aprendizagem supervisionada a não supervisionada é que aqui a aprendizagem ocorre em dados não rotulados, ou seja, não dizemos ao computador o que é aquela entrada. Dado um *dataset* de uma distribuidora que quer classificar seus clientes em categorias, o algoritmo irá categorizar estes em grupos semelhantes, a partir disso poderemos atribuir uma característica a um dos grupos distintos, por exemplo clientes que compram produtos frescos.

Figura 4 – Exemplo de conjunto de dados submetido a algoritmo não supervisionado.



Fonte: Adrian Rosebrock, k-NN classifier for image classification, **pyimagesearch**, 2016 Disponível em: < https://www.pyimagesearch.com/2016/08/08/k-nn-classifier-for-image-classification/ > Acesso em: 27 de março de 2021.

Dado o conjunto de imagens e um número de classes que o algoritmo deve agrupar as características semelhantes, espera-se que o algoritmo caracterize cada imagem corretamente, no exemplo da Figura 4, o número de classes deveria ser dois, representando gatos e cachorros, o algoritmo então categorizaria as imagens com base nas semelhanças.

Dado um conjunto de imagens sem nenhuma indicação explicita, o algoritmo agrupará as características semelhantes em uma quantidade de grupos informada a priori pelo desenvolvedor.

O intuito desta área de pesquisa no campo de sistemas inteligentes, parte do pressuposto da teoria de qualquer conhecimento humano onde, a partir de experiências o conhecimento é formado. Neste contexto, podemos deduzir então que os algoritmos de *machine Learning* são aprendizes, que a

partir de uma experiência com um conjunto de dados serão capazes de generalizar exemplos/tarefas não vistos.

2.4. Métricas de Desempenho

Para analisar o desempenho de um modelo de *machine learning* podemos utilizar métricas definidas na literatura associadas a Tabela de Confusão [Tabela 1], onde é possível verificar a performance do resultado do algoritmo. Por exemplo, abaixo podemos observar uma tabela 1 representando visualmente uma tabela de confusão.

Tabela 1 – Tabela de confusão.

		VALOR REAL	
		GATO	CACHORRO
VALOR PREVISTO	GATO	7	9
	CACHORRO	4	12

Fonte: Autoria própria.

A matriz tem como função representar a taxa de acertos de um determinado algoritmo, os números em verde representam os objetos que o algoritmo apontou corretamente, os em vermelho aponta os que ele errou, cada um destes estados possui um nome, que será apresentado abaixo.

Tabela 2 – Estados de uma predição.

Teste: quais Objetos são azuis ?

Objetos : 

Objeto	Resultado do Teste	Avaliação	O que é ?
	Azul	Correto	True positive
	Azul	Errado	False Positive
	Não Azul	Errado	False negative
	Não Azul	Correto	True Negative

Fonte: Autoria própria.

Assim como na Tabela 2 podemos observar e determinar os resultados da seguinte maneira:

True Positive: O modelo acertou na predição.

False Positive: O modelo errou na predição.

False Negative: O modelo errou na negação do objeto.

True Negative: O modelo acertou na negação do objeto.

Tabela 3 – Matriz de Confusão aplicada ao exemplo.

		VALOR REAL	
		AZUL	NÃO AZUL
VALOR PREVISTO	AZUL	1	1
	NÃO AZUL	1	1

Fonte: Autoria própria.

A partir dos resultados obtidos acima, podemos utilizar as seguintes fórmulas para calcular as métricas de desempenho de um determinado algoritmo de *machine learning* utilizado.

***Acurácia*:** Indica a desempenho geral do modelo dentre as classificações.

Figura 5 – Fórmula 1: Cálculo de acurácia.

$$\frac{VP + VN}{VP + VN + FP + FN} \times 100 \quad (1)$$

Fonte: Métricas de avaliação: acurácia, precisão, recall. Quais as diferenças? **Vitorborbarodrigues**. 2019. Disponível em: <https://vitorborbarodrigues.medium.com/métricas-de-avaliação-acurácia-precisão-recall-quais-as-diferenças-c8f05e0a513c>. Acesso em 30 de maio de 2021.

Precisão: Representa a taxa de classificações corretas de um determinado objeto.

Figura 6 – Cálculo de Precisão.

$$\frac{VP}{VP + FP} \times 100 \quad (2)$$

Fonte: Métricas de avaliação: acurácia, precisão, recall. Quais as diferenças? **Vitorborbarodrigues**. 2019. Disponível em: <https://vitorborbarodrigues.medium.com/métricas-de-avaliação-acurácia-precisão-recall-quais-as-diferenças-c8f05e0a513c>. Acesso em 30 de maio de 2021.

Revocação: Representa a taxa onde o algoritmo errou ao classificar.

Figura 7 – Cálculo de Revocação.

$$\frac{TP}{TP + FN} \times 100 \quad (3)$$

Fonte: Métricas de avaliação: acurácia, precisão, recall. Quais as diferenças? **Vitorborbarodrigues**. 2019. Disponível em: <https://vitorborbarodrigues.medium.com/métricas-de-avaliação-acurácia-precisão-recall-quais-as-diferenças-c8f05e0a513c>. Acesso em 30 de maio de 2021.

2.5. Deep Learning

O *Deep Learning* tem como principal função treinar computadores para realizar tarefas normalmente atribuídas a seres humanos, o que inclui reconhecimento de fala, identificação de imagens e previsões. Porém, de maneira divergente os tipos habituais que utilizam equações predefinidas, o *deep learning* configura parâmetros básicos sobre os dados e treina o computador para aprender sozinho através do reconhecimento de padrões em várias camadas de processamento, para que com isso o computador tenha a capacidade de classificar, reconhecer, detectar e descrever em outras palavras compreender.

Algoritmos de *deep learning* são baseados em representações distribuídas, a suposição subjacente por trás de representações distribuídas é de que os dados observados são gerados pelas inteirações de fatores

organizados em camadas, conclui-se então que a suposição de que essas camadas de fatores correspondem a níveis de abstração ou de composição

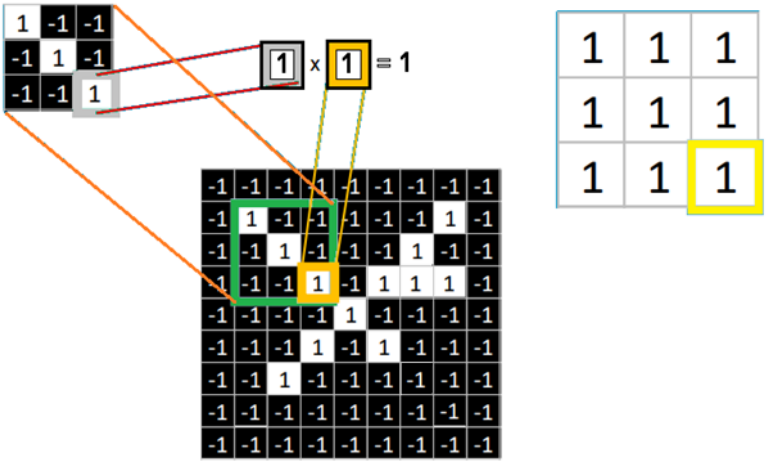
O *deep learning* então aborda diferentes observações por exemplo uma imagem, e as abstrai de modo a tentar reconhecer padrões, uma imagem por exemplo pode ser representada de várias maneiras, tais como um vetor de valores de intensidade por pixel, ou ainda mais abstrato como um conjunto de arestas, os algoritmos [AP].

2.6. Rede Neural Convulacional

Uma rede neural convulacional dentro do contexto de inteligência artificial e aprendizagem de máquina consiste em uma rede neural artificial *feed-forward* que obtém êxito no processamento e análise de imagens digitais [ADI]. Estas redes são inspiradas em processos biológicos onde o padrão de conectividade entre os neurônios baseia-se no córtex visual dos animais, neurônios corticais individuais respondem a estímulos apenas em regiões restritas do campo de visão conhecidas como campos receptivos, os campos receptivos de diferentes neurônios se sobrepõem parcialmente de forma a cobrir todo o campo de visão, de modo que a imagem é processada após a identificação de objetos separadamente e não como uma unidade concreta, o que no campo de estudos relacionados a visão computacional para sistemas já foi testado e cientificamente comprovado que necessita de mais processamento e oferece resultados muitas vezes inferiores a utilização da redes convulsionais [RNC].

Convolução matematicamente falando consiste em uma operação efetuada em duas funções de modo a produzir uma terceira função que normalmente é vista como uma versão modificada de uma das funções originais. No contexto de redes convulsionais, ela é utilizada em um input que para nossa aplicação será uma imagem, uma imagem em um nível de abstração baixo consiste em um *array* de pixels, onde cada pixel tem um peso. Para realizarmos a convolução, precisamos de duas funções, e como já mencionado, nossa primeira função será a entrada, a imagem obtida, precisamos então de uma segunda função para realizar a operação matemática, a qual denominamos de filtro. Um filtro consiste em uma característica que pertence a uma classe, por exemplo as rodas de um carro.

Figura 8 – Exemplo de uma convolução Imagem X Kernel.

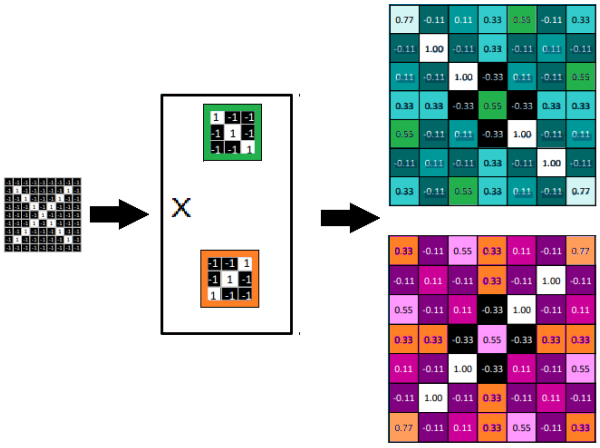


Fonte: EVANDRO SESTREM, **Um estudo de Redes Neurais Convolucionais e sua aplicação na detecção da malária.** Disponível em: <https://medium.com/@sestrem/um-estudo-de-redes-neurais-convolucionais-e-sua-aplicação-na-detecção-da-malária-7dfc7a4b960a>. Acesso em: 05 de julho de 2021.

É feito então uma convolução (Filtro) X (Imagem) e o produto desta operação consiste em uma divisão de X e Y do filtro.

Cada operação gera um valor que constitui uma matriz resultante, esta matriz é denominada mapa de ativação [MA] como pode ser visto na Figura 10 a seguir.

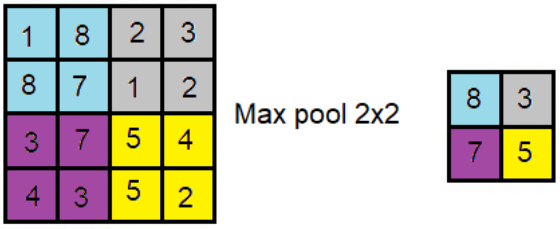
Figura 9 – Exemplo da geração de mapas de ativação.



Fonte: EVANDRO SESTREM, **Um estudo de Redes Neurais Convolucionais e sua aplicação na detecção da malária.** Disponível em: <https://medium.com/@sestrem/um-estudo-de-redes-neurais-convolucionais-e-sua-aplicação-na-detecção-da-malária-7dfc7a4b960a>. Acesso em: 05 de julho de 2021

Após isso a matriz resultante poderá ser submetida a uma nova camada chamada de *Pooling* que busca diminuir detalhes, para isso é definido um tamanho da janela que percorrerá a matriz, nela o maior valor inserido dentro da janela será passado adiante para uma nova matriz [NM].

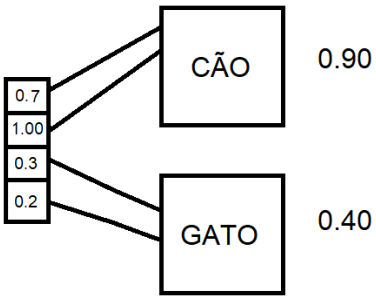
Figura 10 – Exemplo de operação Max Pool.



Fonte: SAVYA KHOSLA, **CNN | Introduction to Pooling Layer.** Disponível em: < https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/>. Acesso em 23 de junho de 2021.

Na imagem 10 podemos observar os resultados obtidos que percorrerá pela matriz resultante de todas estas operações é relacionada com as classes presentes na aplicação com base na proximidade dos valores [PV].

Figura 11 – Exemplo da categorização final de classes de uma rede convulacional.



Fonte: Autoria própria.

Após tudo isso como observado na imagem acima o algoritmo por meio desse observará a diferença entre dois objetos semelhantes e será capaz de saber de qual objeto se trata.

3. TRABALHOS RELACIONADOS

Rosa et.al. [Rosa.Walter. 2016] abordam a problemática do monitoramento de vagas de estacionamento. É descrito no texto que inúmeras formas para solução do problema em questão já foram propostas, é citado explicitamente a dos sensores de presença, porém o alto custo e a frequente necessidade de manutenção inviabilizam a utilização do mesmo. A partir da constatação descrita, os autores sugerem a utilização de conceitos relacionados ao *machine learning* atrelados a arquiteturas de rede sem fio Wireless Sensor Network-(WSN) juntamente a câmeras de segurança Visual Sensor Network-(VSN). A imagem captada através da VSN é submetida a filtros para auxiliar as tomadas de decisão na etapa de processamento. O texto, entretanto, não explicita quais foram as ferramentas utilizadas nos filtros das imagens obtidas. Nosso foco é a utilização da arquitetura VSN, atrelado as melhores operações morfológicas em diferentes condições climáticas.

"*Resultados superiores a outras abordagens acadêmicas*" é o que é descrito por Mexas [Mexas.2014]. No texto o autor aborda os conceitos ligados a operações morfológicas aplicadas a imagens obtidas através de câmeras, que visam a detecção de veículos em vagas de estacionamento para fins de gerenciamento. Exemplos destas técnicas são: erosão, dilatação etc., que visam a alteração da imagem de modo a facilitar o processamento das mesmas. Técnicas de abstração de imagens também são descritas no texto, uma delas é a técnica de detecção de bordas mais conhecida pelo seu nome original de *Canny,* que consiste em abstrair somente as bordas dos objetos contidos na imagem a qual é submetida. O autor passa grande parte de seu trabalho testando diversas combinações de operações visando a que se adeque melhor a sua necessidade, entretanto ele não leva em consideração diferentes condições climáticas. Iremos propor em nosso trabalho as melhores operações morfológicas de acordo com a condição da captura da imagem.

Castro et.al. [R.O 2017] relata em seu texto uma metodologia para gerenciamento de vagas de estacionamento a qual denomina SPANS–Smart Parking Service juntamente a um aplicativo móvel que visa informar o usuário do estacionamento sobre as disponibilidades existentes no espaço, através do processamento proveniente de um nó sensor sem fio que captura

imagens. Estas imagens então, são submetidas a filtros de modo a facilitar seu posterior processamento. No texto, é descrito que foram utilizados um filtro Glaussian Blur para suavização de imagens e *Canny* para abstração de bordas. Para implementação foi utilizada a biblioteca *OpenCV.* Nosso trabalho, entretanto, utilizando-se do conceito de *machine learning* não necessitará de um esforço para delimitação das vagas de estacionamento, isto será solucionado através do treinamento do algoritmo com inúmeras amostras a ele submetido.

Em seu trabalho A. Redondi et.al [Alessandro Redondi 2013] discorre sobre a crescente utilização de paradigmas de análise de imagens, e sua importância para o futuro da tecnologia. Seu foco, entretanto, é o de descobrir qual melhor paradigma para realizar tal tarefa. No trabalho são descritos dois paradigmas: *Compress-then-analyze-*(CTA) e *analyze-then-compress-(*ATC), o primeiro descreve o processo no qual a imagem obtida através de uma câmera é transmitida para um servidor, que será analisada em uma etapa futura do processo, já no ATC a imagem obtida da câmera é manipulada no momento da aquisição, e então transmitida através de uma banda até um servidor. O trabalho, entretanto, em suma, não defini qual o melhor como algo binário, mas sim, afirma que os dois paradigmas atingem seu ápice em diferentes contextos, O CTA atinge os melhores resultados se a largura de banda for grande, e as imagens forem em alta definição, já ATC atinge melhores resultados em larguras de banda inferiores. Portanto, a utilização do paradigma está estritamente ligada ao contexto em que o desenvolvedor poderá utilizar.

4. DESENVOLVIMENTO

Será utilizada uma aplicação capaz de distinguir dentro de um determinado perímetro se há um veículo ocupando uma vaga de estacionamento, recuperando as imagens captadas através de uma câmera e submetendo-as a manipulação. Assim será possível delimitar o comprimento de cada vaga existente e a aplicação também será capaz de identificar objetos da classe carro, pessoas, e outros meios de transporte que podem estar obstruindo a vaga de estacionamento.

4.1 Desenvolvimento da Aplicação Proposta

Uma câmera é utilizada para captação das imagens que serão utilizadas no algoritmo, onde é determinado que a cada período será retirado um *frame* da gravação em tempo real. Esta imagem será então submetida a lógica de representação das vagas e posteriormente a detecção de carros utilizando um algoritmo de inteligência artificial, o resultado será então armazenado em um banco de dados, e posteriormente lido pela aplicação web.

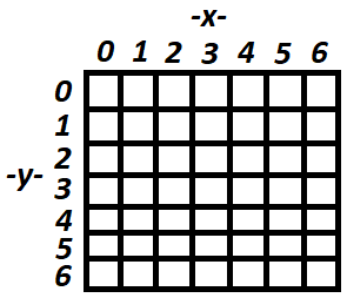
Figura 12 – Representação do sistema.

Fonte: Autoria própria.

- Câmera: Capta a imagem que será utilizada na lógica.
- Estacionamento: Espaço físico que será utilizado na aplicação.
- Lógica: Utilizando aprendizado de máquina juntamente com propriedades do OpenCv será possível delimitar se uma vaga está ou não disponível.
- Banco de dados: Será responsável por guardar o estado das vagas.
- Front-End: Representará ao usuário o estado das vagas.

Ao trabalhar com imagens devemos ter em mente que estamos na verdade trabalhando essencialmente com uma matriz composta de N pixels como representado na figura 13, dentro de nosso projeto a primeira etapa para realização da detecção se a vaga de estacionamento em questão está ou não sendo utilizada, é delimitar o espaço físico que a vaga de estacionamento preenche dentro da imagem, para isso utilizamos a função existente na biblioteca *OpenCv.*

Figura 13 – Representação de uma matriz.



Fonte: Autoria própria.

Essa por sua vez consiste em uma função que recebe 5 parâmetros, sendo eles o primeiro, a imagem, segundo e terceiro as coordenadas dentro da matriz que compõe a imagem como apresentado na imagem 14, a quarta sendo a cor do objeto que será desenhado, e por último a espessura da figura geométrica, que no nosso caso consiste em um retângulo, o código completo pode ser visto na figura 14.

Figura 14 – Função para desenhar vaga.

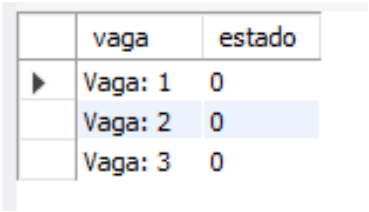
```
cv2.rectangle(image, ref_point[0], ref_point[1], (0, 255, 0), 2)
```

Fonte: Autoria própria.

Utilizando a função da imagem 14, serão armazenadas no banco de dados quantas vagas de estacionamento foram delimitadas no processo. A estrutura do banco de dados consiste em dois campos, sendo o primeiro com o nome de “Vaga” que representa a qual vaga aquela linha pertence, e o segundo com o nome de “Estado” sendo do tipo *booleano*, como representado na imagem 15, onde 0 representa a vaga vazia, e 1 representa

a vaga ocupada. Estas informações serão posteriormente lidas por uma aplicação web que realizará a representação gráfica do estacionamento para o usuário final.

Figura 15 – Banco de dados da aplicação.



	vaga	estado
▶	Vaga: 1	0
	Vaga: 2	0
	Vaga: 3	0

Fonte: Autoria própria.

4.2. Detecção do objeto

Utiliza-se então uma função que será capaz de detectar se um determinado objeto está ou não presente na imagem passada por parâmetro, imagem esta que será obtida a cada intervalo de tempo proveniente da câmera, a função com nome de "*detectCustomObjectsFromImage*" recebe três parâmetro. Primeiro a imagem que será analisada, o segundo o resultado da imagem com os objetos encontrados, e por fim uma taxa de acerto que o algoritmo deve levar em conta, Na Figura 16 qualquer objeto que tenha mais de 30% de chance de pertencer a uma determinada classe será categorizado como ela.

Figura 16 – Código detecção de determinado objeto.

```
cv2.imwrite("frame.png", image)
detections = detector.detectCustomObjectsFromImage(custom_objects=custom,input_image=os.path.join(execution_path , 'frame.png'),
                                                   output_image_path=os.path.join(execution_path , 'frame_detec.png'),
                                                   minimum_percentage_probability=30)
img_result = cv2.imread('frame_detec.png',cv2.IMREAD_COLOR)
```

Fonte: Autoria própria.

O resultado da detecção é inserido em uma cópia da imagem fonte, entretanto para aplicarmos nossa lógica de detecção do projeto e descobrir se existe ou não um veículo estacionado na vaga deve-se manipular o resultado da função, que consiste em um *Array* com as coordenadas onde foi detectada a presença do objeto que escolhemos isto será então guardado em uma variável do tipo *Array*, onde cada posição representa as coordenadas de um carro.

Figura 17 – Representação da lógica para detectar se existe carro na vaga.

Fonte: Autoria própria.

Neste momento já possuímos as duas principais informações para detecção do estado da vaga de estacionamento, onde na imagem obtida é delimitado a vaga, e onde existem possíveis veículos. Realizou-se então uma função lógica, que tem como objetivo definir se dentro das coordenadas que delimitamos como uma vaga de estacionamento, existem carros, sendo r1 correspondente as coordenadas de vagas e r2 coordenadas dos veículos.

Figura 18 – Lógica utilizada para detectar estado vaga.

```
r1.x1 >= r2.x1 and r1.x2 >= r2.x2 and r1.x3 <= r2.x3 and r2.x4 >= r1.x4
```

Fonte: Autoria própria.

Se o retorno da lógica da Figura 19 for igual a verdadeiro, a afirmação de que dentro do espaço que delimitamos uma vaga de estacionamento existe um carro está correta, portanto, a vaga encontra-se ocupada, devemos então atualizar o campo que representa o estado da vaga alterando-o para 1.

Figura 19 – Código python que atualiza banco de dados.

```
for k in arrayVagasOcupadas:
    cursor.execute("UPDATE vagas SET estado = 1 WHERE vaga = '%s'"% (k))
```

Fonte: Autoria própria.

A partir da lógica acima representada o banco sofrerá uma alteração sendo posteriormente lida pela aplicação web, que sinalizará ao usuário que a

vaga está ocupada com a mudança da cor da respectiva representação da vaga.

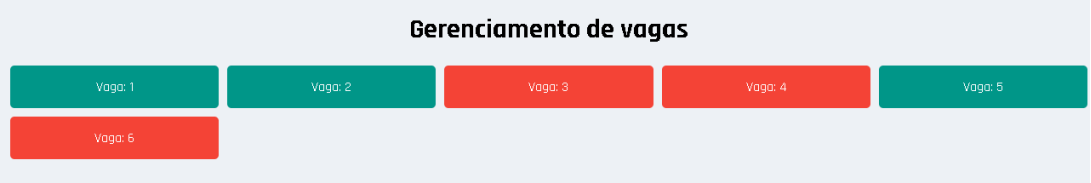
Figura 20 – Banco de dados após atualização.

	vaga	estado
▶	Vaga: 1	0
	Vaga: 2	1
	Vaga: 3	0

Fonte: Autoria própria.

Após a atualização no banco de dados, esse resultado será mostrado na parte do *front-end* que atualizará junto com o banco de dados a cada 5 segundos. Em forma que mostre ao usuário as opções de vagas disponíveis que será apresentado no formado de retângulos verdes e vagas ocupadas apresentadas como retângulos vermelhos.

Figura 21 – Front-End.



Fonte: Autoria própria.

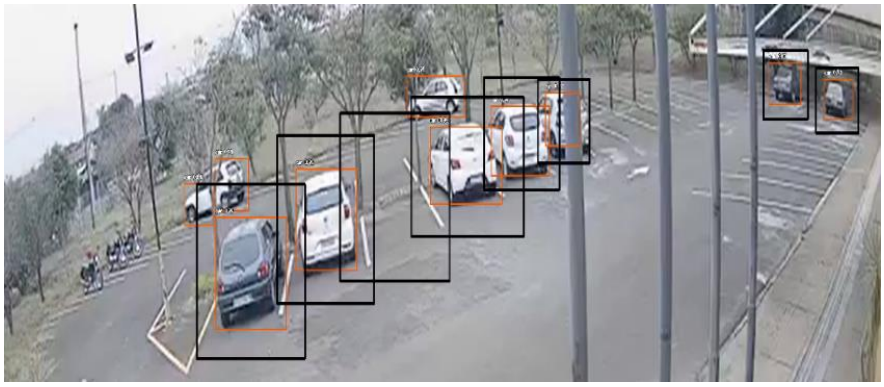
A parte do *front-end* foi desenvolvida no Visual Studio Code [VSC] que é um ambiente de para o desenvolvimento de programação em *JavaScript, CSS e HTML.*

Para auxiliar em sua criação também foi utilizado o *software* Node.js que é uma plataforma de aplicação, na qual se escreve programas com *Javascript.* Trata-se de uma tecnologia assíncrona que trabalha em um único *thread* de execução, foi utilizado para a criação de uma *API* que conecta o *front-end* com banco de dados, assim permitindo que o *front-end* leia o banco de dados e obtenha uma resposta sincronizada, atualizando a um determinado tempo.

Dentro da interface foi criada uma declaração para sempre mostrar o status, tanto sendo “Ocupado” ou “Livre”, dependendo sempre do resultado informado pelo banco de dados, assim a cada atualização também mudará a resposta dada ao *front-end.*

5. Resultados Obtidos

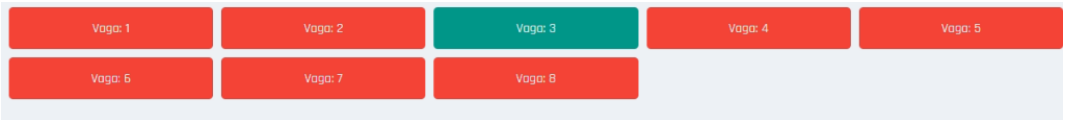
Figura 22 – Sistema em funcionamento



Fonte: Autoria própria

Pode-se observar na Figura 22 a solução proposta no projeto em um ambiente real, os retângulos desenhados em laranja representam o trabalho do algoritmo de inteligência artificial que busca identificar objetos do tipo carro. Já os retângulos desenhados na cor preta representam cada vaga de estacionamento presente no ambiente, tendo estas duas informações previamente podemos então utilizar a lógica descrita no projeto para identificar se o carro está ou não utilizando determinada vaga. Caso a resposta seja afirmativa o sistema irá atualizar o banco de dados alterando o estado da vaga para em uso.

Figura 23 – Sistema web em funcionamento.



Fonte: Autoria própria.

A aplicação web será responsável por representar para o usuário quais as vagas estão disponíveis no momento, isso será possível graças a leitura em tempo real do banco de dados que será atualizado também em tempo real pela aplicação que contêm a inteligência artificial.

6. CONCLUSÃO

Neste trabalho foi desenvolvida uma aplicação uma aplicação capaz de atualizar o estado de uma vaga de estacionamento sem a necessidade de intervenção humana no processo, para que com isso o motorista de um veículo tenha conhecimento prévio da disponibilidade de uma determinada vaga.

Isso foi possível graças aos estudos e desenvolvimento de uma aplicação com propriedades de inteligência artificial que fosse capaz de a partir de um treinamento prévio, reconhecer padrões da imagem submetida a ela, retornando, caso exista algum objeto presente em um determinado espaço, se existir a aplicação terá propriedade para categorizá-lo, e a partir disso atualizar o banco de dados que será lido pela aplicação web.

Melhoramos o projeto inicial de Castro et.al [R.O 2017] em vários aspectos, conseguindo realizar uma leitura mais otimizada das imagens utilizando *machine learning* e futuramente melhoraremos em outros aspectos como num desenho automático para identificação de vagas e nos reconhecimentos de outros veículos como motos, bicicletas etc.

Como trabalhos futuros sugerimos uma aplicação que além das capacidades de detecção de carros e a lógica para verificar o estado da vaga, também exista uma detecção do que é uma vaga de estacionamento, uma vez que as vagas de estacionamento utilizadas no presente trabalho são demarcadas de forma manual.

7. REFERÊNCIAS BIBLIOGRÁFICAS

ADRIAN ROSEBROCK, **k-NN classifier for image classification. Pyimagesearch**, 2016. Disponível em: < https://www.pyimagesearch.com/2016/08/08/k-nn-classifier-for-image-classification/ > Acesso em: 27 de março de 2021.

A História do Python. **MindBending**, 2014. Disponível em: <http://mindbending.org/pt/a-historia-do-python>. Acesso em 21 de março de 2021.

Cars Dataset. **ai.stanford.edu**, 2013. Disponível em: <https://ai.stanford.edu/~jkrause/cars/car_dataset.html>. Acesso em: 27 de março de 2021.

CASTRO, Marcelo R. O.; TEIXEIRA, Marcio A.; NAKAMURA, Luis H. V.; MENEGUETTE, Rodolfo I. **Gerenciamento automático de vagas em estacionamentos baseado em redes de sensores sem fio para ITS**. In: WORKSHOP DE COMPUTAÇÃO URBANA (COURB_SBRC), 1. 2017, 1/2017. Anais do I Workshop de Computação Urbana (COURB 2017). Porto Alegre: Sociedade Brasileira de Computação, maio de 2017, ISSN 2595-2706.

Conheça as 5 melhores linguagens de programação para inteligência artificial. **ComputerWorld**, 2018. Disponível em: <https://computerworld.com.br/2018/07/04/conheca-5-melhores-linguagens-de-programacao-para-inteligencia-artificial/>. Acesso em 09 de fev. de 2021.

EVANDRO SESTREM, **Um estudo de Redes Neurais Convolucionais e sua aplicação na detecção da malária.** Disponível em: <https://medium.com/@sestrem/um-estudo-de-redes-neurais-convolucionais-e-sua-aplicação-na-detecção-da-malária-7dfc7a4b960a>. Acesso em: 05 de julho de 2021

HELTON SIMÕES GOMES, **Só 18% das prefeituras têm plano de cidade inteligente no Brasil**, diz pesquisa. **G1**, 2018. Disponível em:

https://g1.globo.com/economia/tecnologia/noticia/so-18-das-prefeituras-tem-planode-cidade-inteligente-no-brasil-diz-pesquisa.ghtml. Acesso em 20 de março de 2021.

Her-Tyan Yeh.; Bing-Chang Chen, Bo-Xun Wang **A City Parking Integration System Combined with Cloud Computing Technologies and Smart Mobile Devices** In: Eurasia Journal of Mathematics, Science & Technology Education, august 2015, ISSN:1305-8215.

Illumination Estimation and Cast Shadow Detection through a Higher-order Graphical Model. **Researchgate**, 2011. Disponível em: <https://www.researchgate.net/figure/Results-for-the-Motorbikes-class-of-the-Caltech101-dataset-A-synthetic-sun-dial-orange_fig4_224254887>. Acesso em 27 de março de 2021.

Métricas de avaliação: acurácia, precisão, recall. Quais as diferenças? **Vitorborbarodrigues**. 2019. Disponível em: <https://vitorborbarodrigues.medium.com/métricas-de-avaliação-acurácia-precisão-recall-quais-as-diferenças-c8f05e0a513c>. Acesso em 30 de maio de 2021.

NOLETO, C. **Framework: o que é, como ele funciona e para que serve?** Disponível em: https://blog.betrybe.com/framework-de-programacao/o-que-e-framework/. Acesso em 13 de abril

O que é Matriz de Confusão? **Datarisk**. Disponível em: <https://ajuda.datarisk.io/knowledge/o-que-é-matriz-de-confusão>. Acesso em 26 de março de 2021.

OpenCV. **Wikipedia**. Disponível em: <https://pt.wikipedia.org/wiki/OpenCV>. Acesso em 15 de abril de 2021.

Poluição dos carros: entenda seus perigos. **eCycle**, 2016. Disponível em: <https://www.ecycle.com.br/4179-poluicao-de-carros>. Acesso em 09 de fev. de 2019.

SAVYA KHOSLA, **CNN | Introduction to Pooling Layer.** Disponível em: < https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/>. Acesso em 23 de junho de 2021

THIAGO REIS. **Com aumento da frota, país tem 1 automóvel para cada 4 habitantes**, 2014. **G1.** Disponível em: <http://g1.globo.com/brasil/noticia/2014/03/com-aumento-da-frota-pais-tem-1-automovel-para-cada-4-habitantes.html>. Acesso em 26 de março de 2021.